Project: Avocados Price Prediction using Machine Learning

Submitted by:
Vanitha D(EBEON1222710226)

ABSTRACT

Avocados seem to be increasingly popular among millennials. It was observed that over 2.6 billion pounds of avocado were consumed in the United States alone in 2020, as opposed to only 436 million pounds consumed in the year 1985, as per Statista. Avocados are seen as a healthy option and are popular for being a good source of "good fats". The fruit can be spread on toast, eaten raw, or even consumed in the form of a shake. Guacamole, which is a Mexican dip, is also made from avocados. Like most other products, the price of avocados fluctuates based on season and supply, which is why it would be beneficial to have a machine learning model to monitor and predict avocado prices.

CONTENTS

TABLE OF CONTENTS

Chapter 1

Introduction:

Avocados seem to be increasingly popular among millennials. It was observed that over 2.6 billion pounds of avocado were consumed in the United States alone in 2020, as opposed to only 436 million pounds consumed in the year 1985, as per Statista. Avocados are seen as a healthy option and are popular for being a good source of “good fats”. The fruit can be spread on toast, eaten raw, or even consumed in the form of a shake. Guacamole, which is a Mexican dip, is also made from avocados. Like most other products, the price of avocados fluctuates based on season and supply, which is why it would be beneficial to have a machine learning model to monitor and predict avocado prices.

Problem Statement:

More awareness of the sales and prices of avocados can benefit the vendors, producers, associations, and companies. Price prediction based on sales would be a good input in the market to determine shifting of produce to locations where the fruit is more in demand or even encouragement of consumption in places where demand is not up to the mark. Due to the uncertainty in the prices, the company is not able to sell their produce at the optimal price. The idea here is to predict future prices based on data collected of past prices based on geographical location, weather changes, and seasonal availability of avocados.

Study of Existing Systems:

- Avocado Price Prediction [EDA and ML]:

  Author: Abhijit Show - In this project, he did the analysis part and applied machine learning algorithms to predict the price of an avocado.

- Avocado Price Prediction [EDA and ML]:

  Author: Rohit negi -In this project, he also did the analysis part and applied machine learning algorithms to predict the price of an avocado.

Identification of gaps in existing systems:

- Avocado Price Prediction - Author: Abhijit Show - In this project, he applied only three machine learning algorithms to predict the price of an avocado.
- Avocado Price Prediction - Author: Rohit negi -In this project, he also applied only three machine learning algorithms to predict the price of an avocado.

Proposed Solution:

We can apply additional one more Machine learning algorithms Extra Tree Regressor and compare the best model by using the accuracy of each model and we can generate the Feature Importance to understand which feature is used to make key decisions.

Tools/Technology used to implement proposed solution:

- Python
- Pandas
- Numpy
- Matplotlib
- Seaborn
- Plotly
- Sklearn
- Jupyter Notebook

Chapter 2

Features & Predictor

- Date
- Total Volume
- 4046
- 4225
- 4770
- Total Bags
- Small Bags
- Large Bags
- XLarge Bags
- type
- year
- region

Response Variable:

- Average Price

Note:

- Total - 13 columns
- Numerical – 10 - Continuous: Which is quantitative data that can be measured.
- String – 3- Ordinal Data: Categorical data that has an order to it.

Chapter 3

Methodology

Data Cleaning and Pre-processing:

The datasets which were collected from Zomato Restaurant Rating Dataset from Kaggle website contain unfiltered data which must be filtered before the final data set can be used to do analysis. Also, data has some categorical variables which must be modified into numerical values for which we used Panda's library of Python. In data cleaning step, first we checked whether there are any missing or junk values in the dataset for which we used the is null () function.

Machine Learning Algorithms:

a) Extra Tree Regressor:

An extra-trees regressor, this class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

b) Random Forest Regressor:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

c) Decision Tree Regressor:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferredfrom the data features. A tree can be seen as a piecewise constant approximation

d) Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used. There are many names for a regression's dependent variable. It may be called an outcome variable, criterion variable, endogenous variable, or regressand. The independent variables can be calledexogenous variables, predictor variables, or regressors.

Implementation Steps:

As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools like PyCharmor Visual studio for implementing ML algorithms. The advantage of Jupyter is that while writing code, it's really helpful for Data visualization andplotting some graphs like histogram and heatmap of correlated matrices. Let's revise implementation steps: a) Dataset collection. b) Importing Libraries: NumPy, Pandas, Matplotlib, Seaborn and Sklearn libraries were used. c) Exploratory data analysis: For getting more insights about data. d) Data cleaning and pre-processing: Checked for null and junk values using isnull() andisna().sum() functions of python. In Pre-processing phase, we did feature engineering on our dataset. As we converted categorical variables into numerical variables using Pandas library. All our datasets have some categorical variables.

Libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model_selection import train_test_split,cross_val_score

from sklearn.metrics import r2_score

from sklearn.linear_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

fromsklearn.ensembleimportRandomForestRegressor

from sklearn.ensemble import  ExtraTreesRegressor


import warnings

warnings.filterwarnings('ignore')
```

Importing the Dataset and assigning that to the variable df:

In [2]:
```
df = pd.read_csv('Avacado price.csv')
```

In [3]:
```
df
```

Out[3]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2	13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	3	06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	4	29-11-2015	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	7	04-02-2018	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	organic	2018	WestTexNewMexico
18245	8	28-01-2018	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico
18246	9	21-01-2018	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico
18247	10	14-01-2018	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico
18248	11	07-01-2018	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico

18249 rows × 14 columns

Accessing the first 5 rows of a dataframe:

Access the first 5 rows of a dataframe

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2	13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	3	06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	4	29-11-2015	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

Accessing the last 5 rows of a dataframe:

Access the last 5 rows of a dataframe

In [5]:
```
df.tail()
```

Out[5]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
18244	7	04-02-2018	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	organic	2018	WestTexNewMexico
18245	8	28-01-2018	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico
18246	9	21-01-2018	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico
18247	10	14-01-2018	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico
18248	11	07-01-2018	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico

To find out the number of rows and columns :

To find out the number of records and features in the dataset

In [6]:
```
df.shape
```

Out[6]:
```
(18249, 14)
```

Observation:

- There are 18249 records and 14 features in the dataset.

Checking the feature names:

Checking the fearture names

```
In [7]: df.columns

Out[7]: Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046', '4225',
               '4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type',
               'year', 'region'],
              dtype='object')
```

Checking the datatype for each column

Checking the data type for each column

```
In [8]: df.dtypes

Out[8]: Unnamed: 0        int64
        Date             object
        AveragePrice    float64
        Total Volume    float64
        4046            float64
        4225            float64
        4770            float64
        Total Bags      float64
        Small Bags      float64
        Large Bags      float64
        XLarge Bags     float64
click to scroll output; double click to hide   object
        year              int64
        region           object
        dtype: object
```

Observation:

* From the above results,

* Date and region - object datatype.

* Average Price, Total Volume, 4046, 4225, 4770, Total Bags, Small Bags, Large Bags and XLarge Bags -float datatype

* year- int datatype

Information about the dataframe:

Information about the DataFrame

```
In [9]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    18249 non-null  int64  
 1   Date          18249 non-null  object 
 2   AveragePrice  18249 non-null  float64
 3   Total Volume  18249 non-null  float64
 4   4046          18249 non-null  float64
 5   4225          18249 non-null  float64
 6   4770          18249 non-null  float64
 7   Total Bags    18249 non-null  float64
 8   Small Bags    18249 non-null  float64
 9   Large Bags    18249 non-null  float64
 10  XLarge Bags   18249 non-null  float64
 11  type          18249 non-null  object 
 12  year          18249 non-null  int64  
 13  region        18249 non-null  object 
dtypes: float64(9), int64(2), object(3)
memory usage: 1.9+ MB
```

Observation:

* It gives the every information about the dataframe like column name, datatypes, etc.,

Drop the unnecessary column:

In [10]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
df
```

Out[10]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	29-11-2015	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	04-02-2018	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	organic	2018	WestTexNewMexico
18245	28-01-2018	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico
18246	21-01-2018	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico
18247	14-01-2018	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico
18248	07-01-2018	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico

18249 rows × 13 columns

Checking for null values:

To check if there is any null values present in the dataset.

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
```

Observation:

- This dataset does no contain any null values so we can proceed further easily with this dataset.

Checking for duplicate values:

Checking if there is any duplicate values present in the dataset

In [12]:
```
df.duplicated().sum()
```

Out[12]: 0

Observation:

- Fortunately we are having a clean dataset and there is no duplicate values in this dataset.

Renaming the feature names:

Renaming feature names for better understanding

In [46]:
```python
df.rename(columns={'4046':'Small_Hass',
            '4225':'Large_Hass',
            '4770':'Extralarge_Hass'},inplace=True)
df.head()
```

Out[46]:

	Date	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region	Month	Day
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany	12	27
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany	12	20
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany	12	13
3	2015-06-12	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany	6	12
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany	11	29

Checking the unique value for type feature:

Checking the unique value for 'type' feature

In [15]:
```python
df.type.unique()
```

Out[15]:
```
array(['conventional', 'organic'], dtype=object)
```

Observation:

- It has two types one is 'conventional' and the other is 'organic'.

Statistical Summary:

Statistical Summary of the dataset

In [16]:
```python
df.describe()
```

Out[16]:

	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	year
count	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	18249.000000	18249.000000
mean	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.396392e+05	1.821947e+05	5.433809e+04	3106.426507	2016.147899
std	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.862424e+05	7.461785e+05	2.439660e+05	17692.894652	0.939938
min	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	2015.000000
25%	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.088640e+03	2.849420e+03	1.274700e+02	0.000000	2015.000000
50%	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.974383e+04	2.636282e+04	2.647710e+03	0.000000	2016.000000
75%	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.107834e+05	8.333767e+04	2.202925e+04	132.500000	2017.000000
max	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.937313e+07	1.338459e+07	5.719097e+06	551693.650000	2018.000000

Obseration:

- Average Prices seems normally distributed as mean and median are closure to each other.
- Data at Total Volume, Avocado Types (4046, 4225, 4770), Total Bags (Small, Large, XLarge) seems highly skewed Right side (positive skewed)

Correlation Analysis:

Correlation Analysis

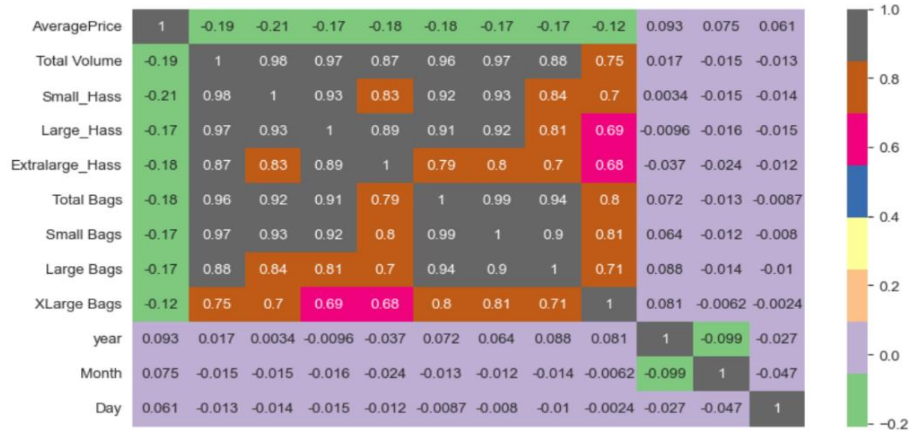
In [17]:
```
df.corr()
```

Out[17]:

	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	year
AveragePrice	1.000000	-0.192752	-0.208317	-0.172928	-0.179446	-0.177088	-0.174730	-0.172940	-0.117592	0.093197
Total Volume	-0.192752	1.000000	0.977863	0.974181	0.872202	0.963047	0.967238	0.880640	0.747157	0.017193
Small_Hass	-0.208317	0.977863	1.000000	0.926110	0.833389	0.920057	0.925280	0.838645	0.699377	0.003353
Large_Hass	-0.172928	0.974181	0.926110	1.000000	0.887855	0.905787	0.916031	0.810015	0.688809	-0.009559
Extralarge_Hass	-0.179446	0.872202	0.833389	0.887855	1.000000	0.792314	0.802733	0.698471	0.679861	-0.036531
Total Bags	-0.177088	0.963047	0.920057	0.905787	0.792314	1.000000	0.994335	0.943009	0.804233	0.071552
Small Bags	-0.174730	0.967238	0.925280	0.916031	0.802733	0.994335	1.000000	0.902589	0.806845	0.063915
Large Bags	-0.172940	0.880640	0.838645	0.810015	0.698471	0.943009	0.902589	1.000000	0.710858	0.087891
XLarge Bags	-0.117592	0.747157	0.699377	0.688809	0.679861	0.804233	0.806845	0.710858	1.000000	0.081033
year	0.093197	0.017193	0.003353	-0.009559	-0.036531	0.071552	0.063915	0.087891	0.081033	1.000000

Visualizing the correlation of feature using heat map

In [50]:
```
plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),annot=True,cmap='Accent')
plt.show()
```



Observation:

* The average price is highly correlated with day,month,year when compared with other features.

Detecting the Outliers:

Detecting Outliers

In [19]:
```
d = df.copy()
d.head(2)
```

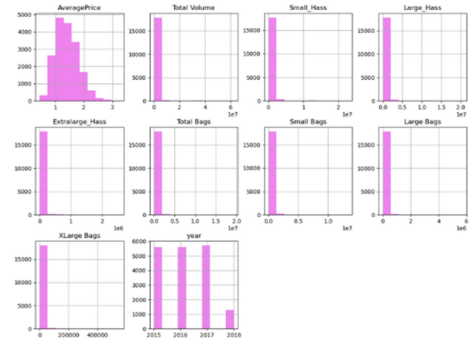
Out[19]:

	Date	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany

Observation:

- We are creating a copy of the dataset to detect and treat the outliers.

In [20]:
```python
df.hist(figsize=(14,14),grid=True,layout=(4,4),color='violet')
plt.show()
```



Log1P method:

In [21]:
```python
d.skew()
```

Out[21]:
```
AveragePrice        0.580303
Total Volume        9.007687
Small_Hass          8.648220
Large_Hass          8.942466
Extralarge_Hass    10.159396
Total Bags          9.756072
Small Bags          9.540660
Large Bags          9.796455
XLarge Bags        13.139751
year                0.215339
dtype: float64
```

Using Log1p Method:

- Numpy log() function is used to get the natural logarithm of value x+1, where x is an element of an array or x is an object.
- The best skew value for normally distributes is very close to zero, so we are using "log1p" method to make the skew value near to zero

In [22]:
```python
skew=('Total Volume','Small_Hass', 'Large_Hass','Extralarge_Hass','Total Bags','Small Bags','Large Bags','XLarge Bags')
for col in skew :
    if d.skew().loc[col]>0.55:
        d[col]=np.log1p(d[col])
```

In [23]:
```python
d.skew()
```

Out[23]:
```
AveragePrice       0.580303
Total Volume       0.088098
Small_Hass        -0.328195
Large_Hass        -0.486654
Extralarge_Hass    0.099986
Total Bags        -0.218874
Small Bags        -0.622148
Large Bags        -0.547765
XLarge Bags        1.176494
year               0.215339
dtype: float64
```

Removing Outliers:

Removing Outliers using Z-Score

In [24]:
```python
from scipy.stats import zscore
z =np.abs(zscore(d['AveragePrice']))
print(z)
print(np.where(z<3))
dn=d[(z<3)]
print('Shape of New Dataframe dn:',dn.shape)
```

```
0        0.188689
1        0.139020
2        1.182069
3        0.809551
4        0.312861
           ...
18244    0.556347
18245    0.755023
18246    1.152375
18247    1.301382
18248    0.531512
Name: AveragePrice, Length: 18249, dtype: float64
(array([    0,     1,     2, ..., 18246, 18247, 18248], dtype=int64),)
Shape of New Dataframe dn: (18118, 13)
```

In [25]:
```
z =np.abs(zscore(dn['Large_Hass']))
print(z)
print(np.where(z<3))
dn1=dn[(z<3)]
print('Shape of New Dataframe dn1:',dn1.shape)
```

```
0        0.370072
1        0.300457
2        0.613605
3        0.467775
4        0.294121
           ...
18244    0.880970
18245    0.598017
18246    0.715576
18247    0.647289
18248    0.729651
Name: Large_Hass, Length: 18118, dtype: float64
(array([    0,     1,     2, ..., 18115, 18116, 18117], dtype=int64),)
Shape of New Dataframe dn1: (18038, 13)
```

In [26]:
```
z =np.abs(zscore(dn1['Total Bags']))
print(z)
print(np.where(z<3))
dn2=dn1[(z<3)]
print('Shape of New Dataframe dn2:',dn2.shape)
```

```
0        0.511006
1        0.471471
2        0.540137
3        0.690272
4        0.662627
           ...
18244    0.315526
18245    0.482877
18246    0.476716
18247    0.407778
18248    0.367331
Name: Total Bags, Length: 18038, dtype: float64
(array([    0,     1,     2, ..., 18035, 18036, 18037], dtype=int64),)
Shape of New Dataframe dn2: (17994, 13)
```

In [27]:
```
z =np.abs(zscore(dn2['Small Bags']))
print(z)
print(np.where(z<3))
dn3=dn2[(z<3)]
print('Shape of New Dataframe dn3:',dn3.shape)
```

```
0        0.272289
1        0.237050
2        0.298892
3        0.436164
4        0.415281
           ...
18244    0.107538
18245    0.257167
18246    0.239415
18247    0.178314
18248    0.141505
Name: Small Bags, Length: 17994, dtype: float64
(array([    0,     1,     2, ..., 17991, 17992, 17993], dtype=int64),)
Shape of New Dataframe dn3: (17827, 13)
```

In [28]:
```
z =np.abs(zscore(dn3['XLarge Bags']))
print(z)
print(np.where(z<3))
dn4=dn3[(z<3)]
print('Shape of New Dataframe dn4:',dn4.shape)
```

```
0        0.655573
1        0.655573
2        0.655573
3        0.655573
4        0.655573
           ...
18244    0.655573
18245    0.655573
18246    0.655573
18247    0.655573
18248    0.655573
Name: XLarge Bags, Length: 17827, dtype: float64
(array([    0,     1,     2, ..., 17824, 17825, 17826], dtype=int64),)
Shape of New Dataframe dn4: (17825, 13)
```

Feature Engineering:

Feature Engineering

In [29]:
```
df['Date']=pd.to_datetime(df['Date'])
df['Month']=df['Date'].apply(lambda x:x.month)
df['Day']=df['Date'].apply(lambda x:x.day)
df.head()
```

Out[29]:

	Date	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region	Month	Day
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany	12	27
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany	12	20
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany	12	13
3	2015-06-12	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany	6	12
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany	11	29

Observation:

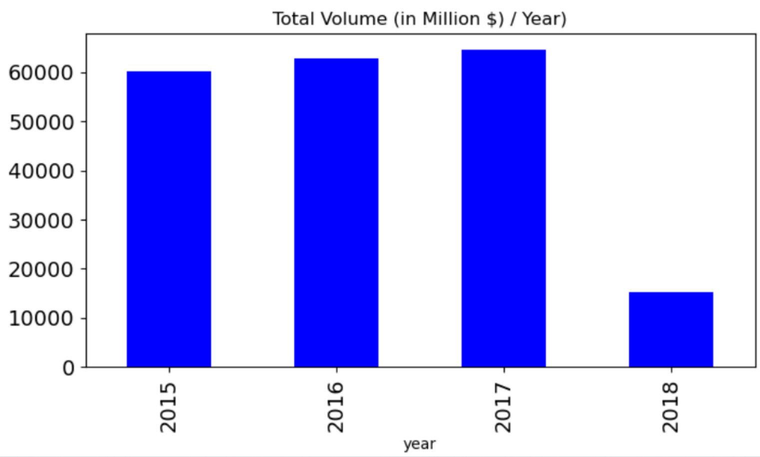
- We are extracting day and month separately from the 'Date' feature using feature engineering for further analysis.

Data Visualization:

Visualizing the Sales of Total Volume by year

In [30]:
```
f,ax = plt.subplots(1,figsize=(8,4))
dn4.groupby(['year'])['Total Volume'].sum().plot(kind='bar', figsize=(8,4), fontsize=14, color='b')
plt.title('Total Volume (in Million $) / Year)')
print('Sales (in Million $):',(df.groupby(['year'])['Total Volume'].sum()/1000000))
plt.show()
```

```
Sales (in Million $): year
2015    4385.468662
2016    4820.889892
2017    4934.305699
2018    1382.738340
Name: Total Volume, dtype: float64
```



Observation:

- Highest Sales recorded in 2017.

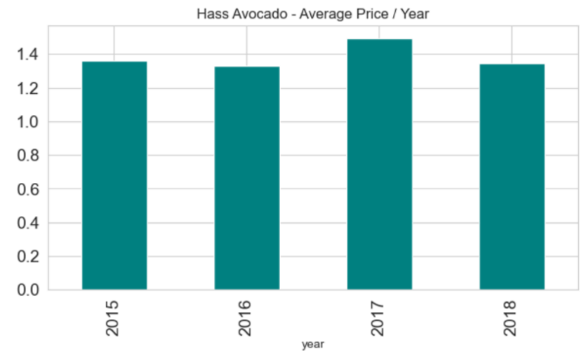
- Sales drastically dropped in 2018.

Visualizing Average Price by Year:

Visualizing Average Price by Year

In [52]:
```
f,ax = plt.subplots(1,figsize=(6,3))
dn4.groupby(['year'])['AveragePrice'].mean().plot(kind='bar', figsize=(8,4), fontsize=14, color='sandybrown')
plt.title('Hass Avocado - Average Price / Year')
print('Average Price(in $):',df.groupby(['year'])['AveragePrice'].mean())
plt.show()
```

```
Average Price(in $): year
2015    1.375590
2016    1.338640
2017    1.515128
2018    1.347531
Name: AveragePrice, dtype: float64
```



Observation:

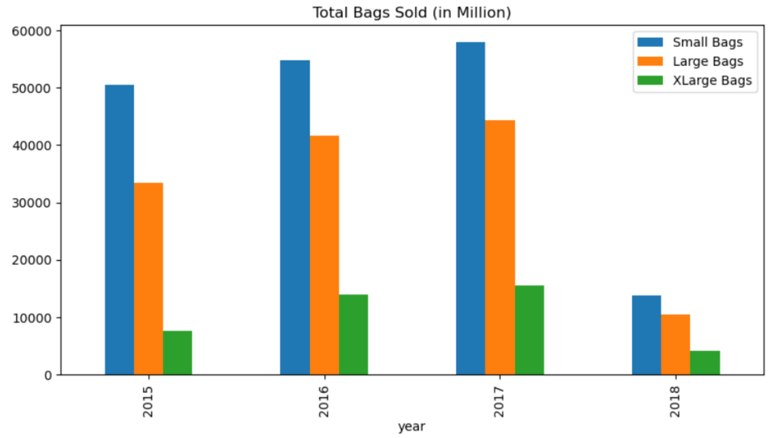
- The average price of avocados (ie.1.57) is high in the year of 2017.

Total Bags sold by Year Wise:

Visualizing the different type of total bags sold by Year

In [33]:
```
d3=d.copy()
d3.drop(['Date','AveragePrice','Total Volume','Small_Hass','Large_Hass','Extralarge_Hass','region','Total Bags'],axis=1,inplace=1
d3.groupby(['year']).sum().plot(kind='bar',figsize=(10,5),legend=True)
plt.title ('Total Bags Sold (in Million)')
print('Total Small Bags sold (in Million):',(df.groupby(['year'])['Small Bags'].sum())/1000000)
print('\n')
print('Total Large Bags sold (in Million):',(df.groupby(['year'])['Large Bags'].sum())/1000000)
```

```
Total Small Bags sold (in Million): year
2015     634.682705
2016    1106.494240
2017    1222.952525
2018     360.741368
Name: Small Bags, dtype: float64


Total Large Bags sold (in Million): year
2015    132.066400
2016    336.626342
2017    399.339040
2018    123.583988
Name: Large Bags, dtype: float64
```



Observation:

- For every year, small bags were mostly bought by people.

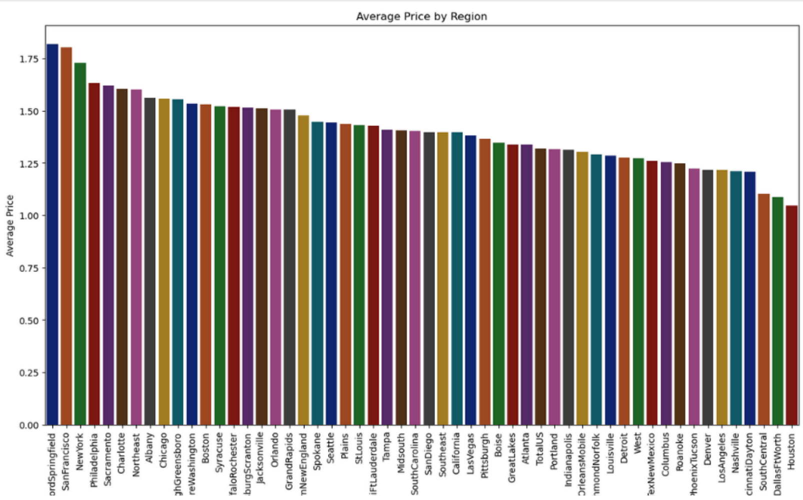
Average Price by Region wise:

Visualizing How Average Price Varies by Region

In [34]:
```
df.region.unique()
```

Out[34]:
```
array(['Albany', 'Atlanta', 'BaltimoreWashington', 'Boise', 'Boston',
       'BuffaloRochester', 'California', 'Charlotte', 'Chicago',
       'CincinnatiDayton', 'Columbus', 'DallasFtWorth', 'Denver',
       'Detroit', 'GrandRapids', 'GreatLakes', 'HarrisburgScranton',
       'HartfordSpringfield', 'Houston', 'Indianapolis', 'Jacksonville',
       'LasVegas', 'LosAngeles', 'Louisville', 'MiamiFtLauderdale',
       'Midsouth', 'Nashville', 'NewOrleansMobile', 'NewYork',
       'Northeast', 'NorthernNewEngland', 'Orlando', 'Philadelphia',
       'PhoenixTucson', 'Pittsburgh', 'Plains', 'Portland',
       'RaleighGreensboro', 'RichmondNorfolk', 'Roanoke', 'Sacramento',
       'SanDiego', 'SanFrancisco', 'Seattle', 'SouthCarolina',
       'SouthCentral', 'Southeast', 'Spokane', 'StLouis', 'Syracuse',
       'Tampa', 'TotalUS', 'West', 'WestTexNewMexico'], dtype=object)
```

In [35]:
```
byRegion=df.groupby('region').mean()
byRegion.sort_values(by=['AveragePrice'], ascending=False, inplace=True)
plt.figure(figsize=(15,8),dpi=100)
sns.barplot(x = byRegion.index,y=byRegion["AveragePrice"],data = byRegion,palette='dark')
plt.xticks(rotation=90)
plt.xlabel('Region')
plt.ylabel('Average Price')
plt.title('Average Price by Region')
plt.show()
```



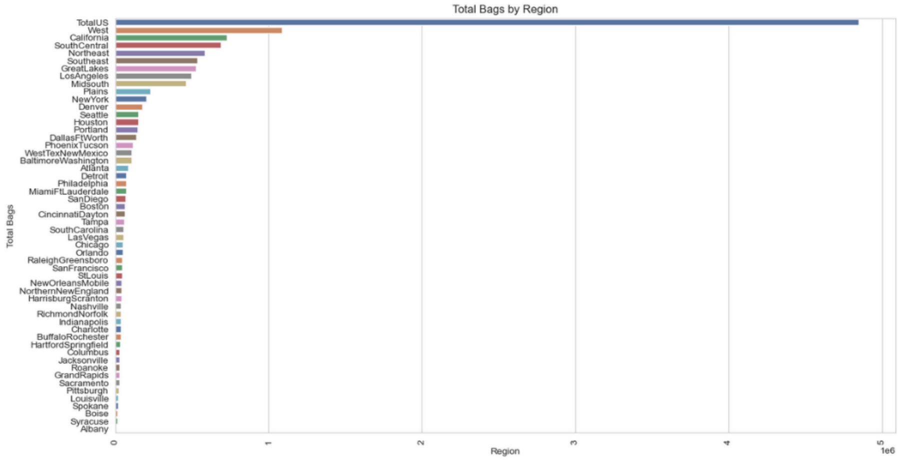
Observation:

- The top 3 regions with highest average price
- HartfordSpringfield
- SanFrancisco
- NewYork

Total Bags By Region wise:

Total Bags by Region wise

```
In [54]: byRegion=df.groupby('region').mean()
byRegion.sort_values(by=['Total Bags'], ascending=False, inplace=True)
plt.figure(figsize=(15,8),dpi=100)
sns.barplot(x=byRegion["Total Bags"],y = byRegion.index,data = byRegion,palette='deep')
plt.xticks(rotation=90)
plt.xlabel('Region')
plt.ylabel('Total Bags')
plt.title('Total Bags by Region')
plt.show()
```



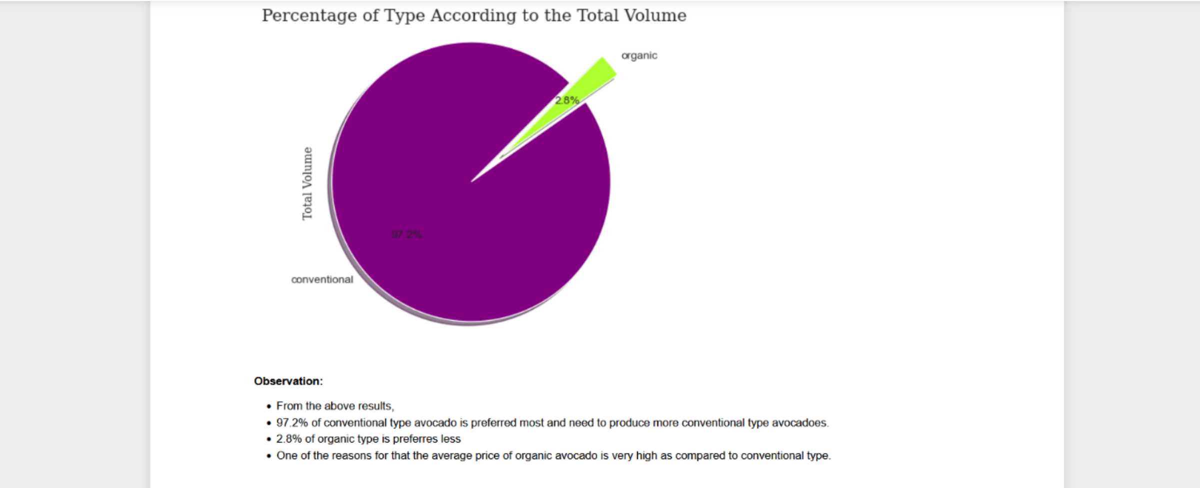
Observation:

- From the above observation, total bags were high in Total US.

Percentage of Type According to Total Volume:

Checking which type of Avocado is preferred most.

```
In [37]: x=df['type'].unique()
x

Out[37]: array(['conventional', 'organic'], dtype=object)

In [45]: x=['conventional','organic']
y=df.groupby('type')['Total Volume'].sum()
sns.set_style('whitegrid')
plt.figure(figsize=(5,5))
colors=['purple','greenyellow']
plt.pie(y,labels=x,shadow=True,colors=colors,explode=(0.3,0),autopct='%1.1f%%',startangle=45)
plt.title('Percentage of Type According to the Total Volume',fontsize=15, fontfamily='serif')
plt.ylabel('Total Volume',fontsize=10, fontfamily='serif',)
plt.axis('equal')
plt.show()
```



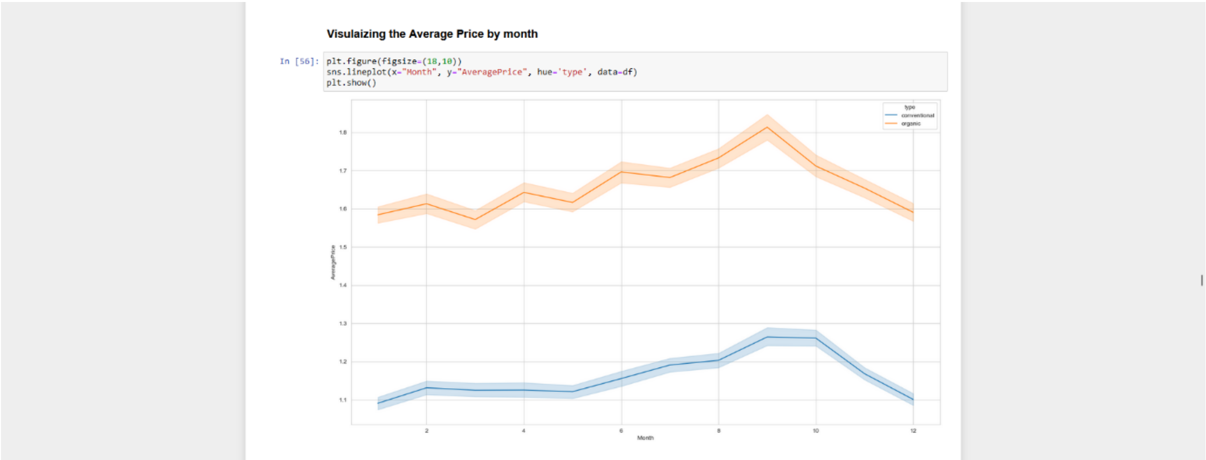
Frequency of Average Price:

Observation:

- The average price is positively skewed
- The minimum average price is 1.10
- The maximum average price is 3.25

Average Price by Date wise:

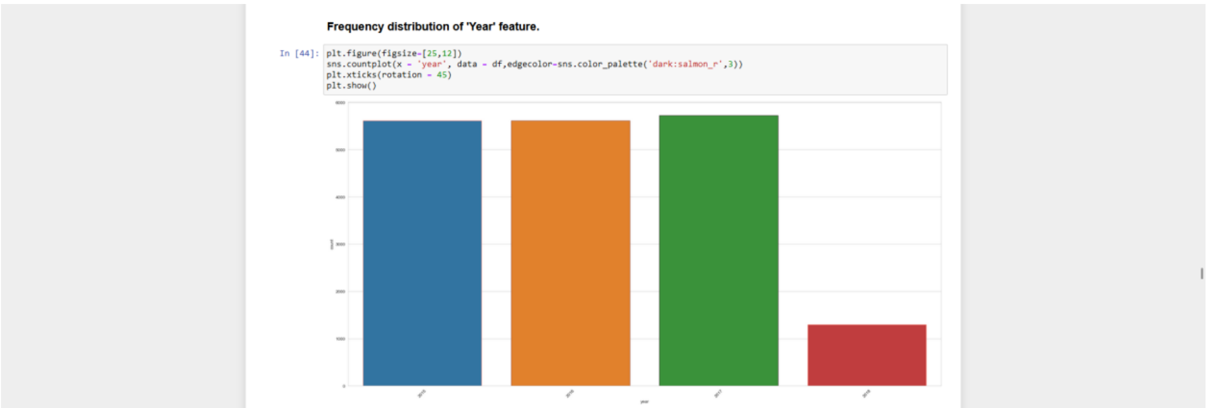
Average Price by Month wise:



Observation:

- We can see that the average price is peak at September month.
- From the observation we can interpret that in September month, the avocados were sold highly.

Frequency Distribution of Year feature:



Observation:

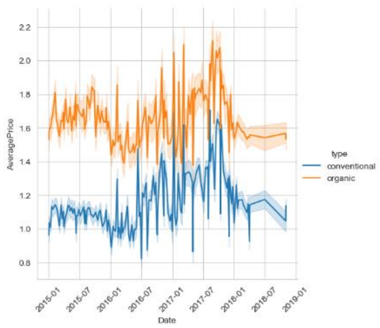
- From the above visualization, we can say that the avocados sold were peak in the year 2017.

Average Price with Date and Type feature:

Visulaizing the Average Price with Date and Type features.

```
In [43]: plt.figure(figsize=(20,10))
sns.relplot(x="Date",y="AveragePrice",hue="type",kind="line",data=df)
plt.xticks(rotation=45)
plt.show()
```

<Figure size 2000x1000 with 0 Axes>



Observation:

- From the above visualization, the average price of conventional type of Hass Avocado is approximately equals to 1.7 in the year 2017-07 and the average price of organic type of Hass Avocado is approximately equals to 2.1 in the same year.
- The average price of organic type of avocado is the highest.

Drop the Date now it is not needed:

Deleting the 'Date' records from the dataset as it is an unnecessary one.

```
In [44]: df.drop(columns=["Date"],inplace=True)
```

Observation:

- Date is not needed now and it is an unecessary column so that it has been dropped.

Handling Categorical Variables:

Handling the Categorical Variables

```
In [45]: le = LabelEncoder()
df['type']= le.fit_transform(df['type'])
df
```

Out[45]:

	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region	Month	Day
0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	2015	Albany	12	27
1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	2015	Albany	12	20
2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	0	2015	Albany	12	13
3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	0	2015	Albany	6	12
4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	0	2015	Albany	11	29
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	1	2018	WestTexNewMexico	4	2
18245	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	1	2018	WestTexNewMexico	1	28
18246	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	1	2018	WestTexNewMexico	1	21
18247	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	1	2018	WestTexNewMexico	1	14
18248	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	1	2018	WestTexNewMexico	7	1

18249 rows × 14 columns

In [46]:
```
df['region'] = pd.Categorical(df['region'])
df_region = pd.get_dummies(df['region'], prefix = 'region')
df_region
```

Out[46]:

	region_Albany	region_Atlanta	region_BaltimoreWashington	region_Boise	region_Boston	region_BuffaloRochester	region_California	region_Charlotte
0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
18244	0	0	0	0	0	0	0	0
18245	0	0	0	0	0	0	0	0
18246	0	0	0	0	0	0	0	0
18247	0	0	0	0	0	0	0	0
18248	0	0	0	0	0	0	0	0

18249 rows × 54 columns

In [47]:
```
df = pd.concat([df, df_region], axis=1)
df.drop(columns="region",inplace=True)
df
```

Out[47]:

	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	type	...	region_SouthCarolina	region_SouthC
0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	...	0	
1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	...	0	
2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	0	...	0	
3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	0	...	0	
4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
18244	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	1	...	0	
18245	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	1	...	0	
18246	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	1	...	0	
18247	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	1	...	0	
18248	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	1	...	0	

18249 rows × 67 columns

Observation:

- Adding the one hot encoded columns for region into our data and dropping the region column from our dataset.

In [48]:
```
df['Month'] = pd.Categorical(df['Month'])
df_month = pd.get_dummies(df['Month'], prefix = 'month')
df_month
```

Out[48]:

	month_1	month_2	month_3	month_4	month_5	month_6	month_7	month_8	month_9	month_10	month_11	month_12
0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
18244	0	0	0	1	0	0	0	0	0	0	0	0
18245	1	0	0	0	0	0	0	0	0	0	0	0
18246	1	0	0	0	0	0	0	0	0	0	0	0
18247	1	0	0	0	0	0	0	0	0	0	0	0
18248	0	0	0	0	0	0	1	0	0	0	0	0

18249 rows × 12 columns

In [49]:
```
df = pd.concat([df, df_month], axis=1)
df.drop(columns="Month",inplace=True)
df
```

Out[49]:

	AveragePrice	Total Volume	Small_Hass	Large_Hass	Extralarge_Hass	Total Bags	Small Bags	Large Bags	XLarge Bags	type	...	month_3	month_4	month_5	month_
0	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	...	0	0	0	
1	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	...	0	0	0	
2	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	0	...	0	0	0	
3	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	0	...	0	0	0	
4	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18244	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	1	...	0	1	0	
18245	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	1	...	0	0	0	
18246	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	1	...	0	0	0	
18247	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	1	...	0	0	0	
18248	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	1	...	0	0	0	

18249 rows × 78 columns

Defining X and Y variables:

Defining x and y variables

```
In [50]: X=df.iloc[:,1:78]                  # defining x and y variables
         y=df['AveragePrice']
```

Splitting by Train and Test

```
In [51]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
         y_test = np.array(y_test,dtype = float)
```

```
In [52]: print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)

         (14599, 77)
         (3650, 77)
         (14599,)
         (3650,)
```

Standard Scaler and Cross Validation:

Normalizing our X_train and X_test using standard scaler ¶

```
In [53]: sc=StandardScaler()
         X_train=sc.fit_transform(X_train)
         X_test=sc.transform(X_test)
```

Cross Validating to improve the accuracy

```
In [54]: def model_accuracy(model,X_train=X_train,y_train=y_train):
             accuracies = cross_val_score(estimator = model, X = X_train, y = y_train, cv = 5)
             print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
             print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

Fitting the Model:

1.Linear Regression:

Model Fitting

Model1: Multiple Linear Regression

1. Are we good with Linear Regression? Lets find out.

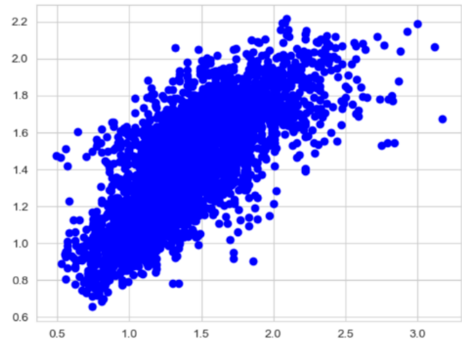
Creating and Fitting the model

```
In [55]: lr=LinearRegression()
         lr.fit(X_train,y_train)
         y_pred=lr.predict(X_test)
         print(y_pred)

         [0.94669972 1.12818585 1.34653388 ... 1.05583058 1.83409177 1.42729054]
```

In [56]:
```
new_df_lr = pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
new_df_lr
```
Out[56]:

	Actual	Predicted
0	1.04	0.946700
1	0.98	1.128186
2	1.39	1.346534
3	1.87	1.871182
4	1.97	1.731276
...	...	...
3645	1.45	1.149116
3646	1.89	1.129798
3647	0.95	1.055831
3648	1.51	1.834092
3649	1.81	1.427291

3650 rows × 2 columns

In [57]:
```
plt.scatter(x=y_test,y=y_pred,color='b')
plt.show()
```


Finding the Accuracy

In [59]:
```
acc_lr=r2_score(y_test,y_pred)*100,2
print(acc_lr)
```
```
(58.43929242002177, 2)
```

Observation:

- By checking the accuracy score, the linear regression model is not the best to apply on our dataset.

Observation:

- The accuracy is 58.43% for linear regression and it won’t be a good model for predicting the average price.

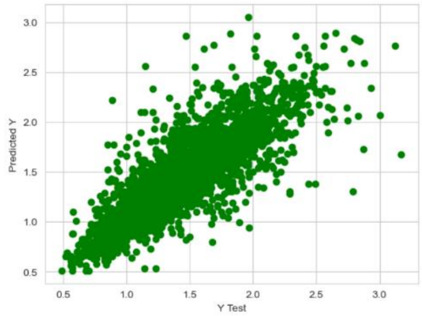
2.Decision Tree Regressor:

Model2: Decision Tree Regressor

2.Are we good with Decision Tree Regressor?

In [60]:
```
dtr=DecisionTreeRegressor(random_state=3)
dtr.fit(X_train,y_train)
pred=dtr.predict(X_test)
```

```
In [61]: plt.scatter(x=y_test,y=pred,color='g')
         plt.xlabel('Y Test')
         plt.ylabel('Predicted Y')
         plt.show()
```




```
In [62]: print('MAE:', mean_absolute_error(y_test, pred))
         print('MSE:', mean_squared_error(y_test, pred))
         print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))

MAE: 0.14400000000000002
MSE: 0.047395835616438356
RMSE: 0.21770584653710692

In [63]: acc_dtr=r2_score(y_test,pred)*100
         acc_dtr

Out[63]: 70.15440738991722
```

Observation:

- The accuracy for Decision Tree Regressor is only 70. This is also not a good model for predicting the price of an avocado.

Observation:

- The accuracy for Decision Tree Regressor is 70%. It is fine but if accuracy is at least 80% would be a great fit.

3.RandomForest Regressor:

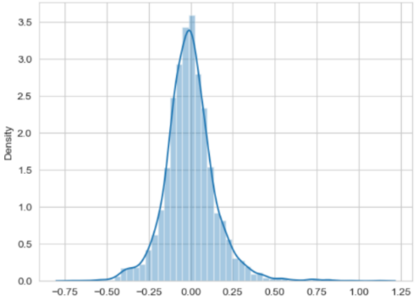
Model3: RandomForest Regressor

```
In [64]: from sklearn.ensemble import RandomForestRegressor
         rdr = RandomForestRegressor(n_estimators=150,max_features='log2',random_state=42)
         rdr.fit(X_train,y_train)
         pred1=rdr.predict(X_test)

In [65]: print('MAE:', mean_absolute_error(y_test, pred1))
         print('MSE:', mean_squared_error(y_test, pred1))
         print('RMSE:', np.sqrt(mean_squared_error(y_test, pred1)))

MAE: 0.10843977625570775
MSE: 0.022968464789628613
RMSE: 0.15155350470915746
```

Observation:

- From the above metrics, we can see the RMSE is lower than the two previous models. So the RandomForest Regressor is the best model in this case.

```
In [66]: sns.distplot((y_test-pred1),bins=50)
         plt.show()
```



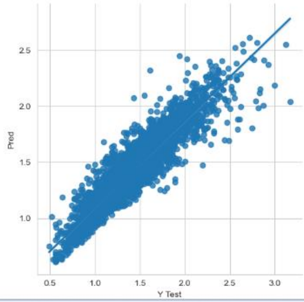
Observation:

- The datas are normally distributed

```
In [67]: data = pd.DataFrame({'Y Test':y_test , 'Pred':pred1},columns=['Y Test','Pred'])
         sns.lmplot(x='Y Test',y='Pred',data=data,palette='viridis')
         data.head()
```

Out[67]:

	Y Test	Pred
0	1.04	1.029667
1	0.98	1.073000
2	1.39	1.407467
3	1.87	1.921933
4	1.97	1.840067




```
In [68]: acc_rdr=r2_score(y_test,pred1)*100
         acc_rdr

Out[68]: 85.53654695450648
```

Observation:

- The accuracy for Randomforest regressor is 85.53%.

4.ExtraTree Regressor:

Model4: Extra Tree Regressor

```
In [69]: etr = ExtraTreesRegressor(n_estimators=100, random_state=0)
         etr.fit(X_train, y_train)

Out[69]: ExtraTreesRegressor(random_state=0)

In [70]: y_predict=etr.predict(X_test)

In [71]: acc_etr=r2_score(y_test,y_predict)*100
         acc_etr

Out[71]: 89.96773398757587
```

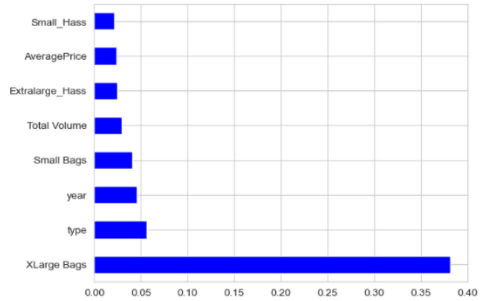
Feature Selection:

Feature Selection

In [81]:
```
importance = etr.feature_importances_

for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.02416
Feature: 1, Score: 0.02952
Feature: 2, Score: 0.02120
Feature: 3, Score: 0.00869
Feature: 4, Score: 0.02495
Feature: 5, Score: 0.01744
Feature: 6, Score: 0.04097
Feature: 7, Score: 0.00454
Feature: 8, Score: 0.38114
Feature: 9, Score: 0.05622
Feature: 10, Score: 0.04530
Feature: 11, Score: 0.00250
Feature: 12, Score: 0.00232
Feature: 13, Score: 0.00369
Feature: 14, Score: 0.00582
Feature: 15, Score: 0.00352
Feature: 16, Score: 0.00680
Feature: 17, Score: 0.00166
Feature: 18, Score: 0.00530
Feature: 19, Score: 0.00349
```

In [82]:
```
index=df.columns[:-1]
importance = pd.Series(etr.feature_importances_,index)
importance.nlargest(8).plot(kind='barh',colormap='winter')
plt.show()
```



Chapter4

Analysis of the Best Results:

- The accuracy for linear regression is 58.43%.
- The accuracy for DecisionTree regressor is 70.15%.
- The accuracy for RandomForest regressor is 85.53%.
- The accuracy for linear regressor is 89.96%.
- From the results, we can get to know that ExtraTree regressor is the best model for predicting the average price of avocado
- which gives the accuracy of 89.96%.

Chapter5

Conclusion:

Conclusion:

- Our ExtraTree Regressor algorithm yields the highest accuracy,89.96%. Any accuracy above 70% is considered good, but be careful because if your accuracy is extremely high, it may be too good to be true (an example of Overfitting). Thus, 80% is the ideal accuracy!
- Out of the 8 features we examined, the top 5 significant features that helped us to predict the price (i.e.) XLArge Bags, type, year, Small Bags,Tota Volume.
- Our machine learning algorithm will able to predict the Avocado Price.
- Based upon the analysis of the result, we can suggest the company owners to enocurage the people to buy organic type of avocadoes.
- The two types are organic and conventional.
- The vendors, companies need to concentrate on those regions in which the avocados are not sold mostly to increase their profit.

Reference:

1. Avocado Price dataset from Kaggle - **JUSTIN KIGGINS**
2. Zomato Restaurant EDA an ML- **Adaikkkappan**

https://github.com/Adaikkkappan/ZomatoRestaurantRating-EDA-and-MachineLearning/blob/main/Zomato%20Restaurant%20Rating.ipynb

3. Avocado Price EDA and ML – **Rohit negi**

https://github.com/rohinegi548/EDA-and-Machine-Learning-Avocado-Prices-Predictions/blob/master/Avocado%20Dataset%20Analysis%20and%20ML%20Predictions.ipynb

4.Avocado Price EDA and ML – **Abijit show**

https://github.com/abhijitshow07/Avocado-Price-Prediction-in-USA/blob/master/avocado_prices.ipynb